Optimization of a Tutoring System from a Fixed Set of Data

Olivier Pietquin, Lucie Daubigney, Matthieu Geist

Optimization of a Tutoring System from a Fixed Set of Data

Olivier Pietquin[1,2]*, Lucie Daubigney*[1]*, Matthieu Geist*[1]

[1]Supélec - Metz Campus, IMS Research Group, France
[2]UMI 2958 (GeorgiaTech - CNRS), France
firstname.lastname@supelec.fr

Abstract

In this paper, we present a general method for optimizing a tutoring system with a target application in the domain of second language acquisition. More specifically, the optimisation process aims at learning the best sequencing strategy for switching between teaching and evaluation sessions so as to maximise the increase of knowledge of the learner in an adapted manner. The most important feature of the proposed method is that it is able to learn an optimal strategy from a fixed set of data, collected with a hand-crafted strategy. This way, no model (neither cognitive nor probabilistic) of learners is required but only observations of their behavior when interacting with a simple (non-optimal) system. To do so, a particular batch-mode approximate dynamic programming algorithm is used, namely the Least Square Policy Iteration algorithm. Experiments on simulated data provide promising results.
Index Terms: Tutoring systems, approximate dynamic programming

1. Introduction

The work described in this paper takes place in the context of a larger project[1] aimed at developing an artificial tutoring system for second language acquisition (especially for French and German languages). In the framework of this project, a serious game environment (I-FLEG) [1] has been designed for interactively learning French as a second language. The game is integrated in Second Life and exploits the 3D virtual reality environment. Thanks to this environment, many important desired features of computer-aided learning technologies can be obtained. Among them, personalization of the learning process is the focus of this paper. In addition, the web-based technology facilitates the collection of big amounts of data that can be used by data-driven methods for optimization.

It has been proven earlier that personalization of the tutoring method is very important in the relationship between teachers and learners [2]. Ideally, each learner should receive adapted courses that allow reaching the best of his/her capabilities. It is natural to think that the ideal situation could be met thanks to computer-aided learning technologies installed on personal computers or accessible through the web. Yet, the actual situation is far from being optimal since current commercial systems are designed for a large public and not for each student. Even worst, they address an average profile which generally simply doesn't exist. It is particulary true in the context of second language acquisition where the errors can be very dependent on the learner because of lexical confusions and pronunciation errors that may be caused by cultural and educational reasons. It is therefore important to design systems that are able to adapt their behavior to the particular profile of the learner. To do so, they have to react online, during their interaction with the learner.

Here, we will suppose that the degree of freedom of the interface stands in the sequencing of teaching and evaluation phases for a specific item to teach. A teaching phase will aim at improving the knowledge of the learner about this item while an evaluation phase will aim at quantifying this knowledge. Thus, in a given situation defined w.r.t. the history of the interaction with the learner, the system will have to take a decision between these two choices. The adaptation takes place at the level of the sequencing of these decisions which should be different from one learner to another. The problem of adapting the system's behavior to the learner can thus be seen as a sequential decision making problem.

In this paper, we propose to solve this problem thanks to a machine learning method (namely Reinforcement Learning (RL) [3]). Especially, we used a batch-mode RL algorithm (namely the Least Square Policy Iteration (LSPI) [4]) to solve this problem from a fixed set of data. Methods of this kind have been recently used successfully in the domain of spoken dialogue systems optimisation [5]. Working off-line on a fixed set of data is an important feature that distinguishes this work from previous attempts to use RL in the context of tutoring systems [6, 7]. Indeed, unlike this work, the method proposed here doesn't require any modeling of the learner or any direct interaction with users while learning. Only data collected with a hand-tuned system are enough or logs of deployed systems. Therefore, the optimisation method doesn't suffers from additional modeling errors occurring when developing a learner simulation method. Indeed, the impact of such models on the quality of the optimized strategies is difficult to predict and some studies even report that totally random models may lead to better strategies [8]. The same problems occur when modeling users for training spoken dialogue systems [9]. Working on fixed sets of data also prevents the real learners to serve as testers of imperfect systems that evolve on-line.

The rest of this paper is organized as follows. Section 2 presents the theoretical background of Reinforcement learning and the LSPI algorithm. Then, Section 3 explains how the tutoring optimisation problem can be cast into the RL paradigm. Preliminary experimental results are presented in Section 4 and show the efficiency of the method on a simplified problem. Finally, Section 5 concludes.

2. Reinforcement Learning

Reinforcement Learning (RL) [3] is a general machine learning paradigm aiming at solving sequential decision making problems. In the RL paradigm, an agent interacts with a system which it tries to control. The system is assumed to be made up of states and the control takes the form of actions performed

[1]ALLEGRO : www.allegro-project.eu

on the system. After each action performed by the agent, the system steps from one state to another and generates an immediate reward which is visible to the agent. The agent's goal is to learn a mapping from states to actions that will maximize some cumulative function of rewards (long-term reward). Therefore, the agent searches for the best sequence of actions and not for actions that are locally optimal.

2.1. Markov Decision Processes

To solve the RL problem described above, the paradigm of Markov Decision Processes (MDP) [10] is traditionally used. An MDP is defined as a tuple $\{S, A, R, P, \gamma\}$, where S is the set of all possible states, A is the set of actions, R is the reward function, P is the set of Markovian transition probabilities and γ is the discount factor. A strategy or a policy π is a mapping from $s \in S$ to $a \in A$. Given the policy π each state can be associated to a value ($V^\pi : S \to \mathbb{R}$) defined as the expected discounted sum of rewards that can be obtained by an agent over the infinite horizon starting from state s and following the policy π:

$$V^\pi(s) = E[\sum_{k=0}^{\infty} \gamma^k r_k | s_0 = s, \pi].$$

This value function quantifies the quality of a given policy; it is what the agent tries to maximize for each state. The state-action value (or Q-) function ($Q^\pi : S \times A \to \mathbb{R}$) adds a degree of freedom for the choice of the first performed action:

$$Q^\pi(s,a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a, \pi] \quad (1)$$

$$Q^*(s,a) = E[\sum_{i=0}^{\infty} \gamma^i r_i | s_0 = s, a_0 = a, \pi^*] \quad (2)$$

$Q^*(s,a)$ is the Q-function of the optimal policy π^* which maximizes the value of each state (or state-action pair):

$$\pi^* = \operatorname*{argmax}_\pi V^\pi = \operatorname*{argmax}_\pi Q^\pi$$

The optimal policy is *greedy* respectively to the optimal state-action value function Q^*:

$$\pi^*(s) = \operatorname*{argmax}_{a \in A} Q^*(s,a).$$

Dynamic programming (DP) [11] aims at computing the optimal policy π^*, using the Q-function as an intermediate and in the case where the transition probabilities and the reward function are known. Especially, the *policy iteration* algorithm computes the optimal policy in an iterative way. An initial policy is arbitrarily set to π_0. At iteration k, the policy π_{k-1} is evaluated, that is the associated Q-function $Q^{\pi_{k-1}}(s,a)$ is computed. To do so, the Markovian property of the transition probabilities is used to rewrite Eq. (1) as :

$$\begin{aligned} Q^\pi(s,a) &= E_{s'|s,a}[R(s,a,s') + \gamma Q^\pi(s', \pi(s'))] \\ &= T^\pi Q^\pi(s,a) \end{aligned} \quad (3)$$

This is the so-called Bellman evaluation equation and T^π is the Bellman evaluation operator. T^π is linear and eq. (3) therefore defines a linear system. It can be solved in an efficient manner by an iterative method using the fact that Q^π is the unique fixed-point of the Bellman evaluation operator (T^π being a contraction). Then the policy is improved, that is π_k is greedy with respect to $Q^{\pi_{k-1}}$:

$$\pi_k(s) = \operatorname*{argmax}_{a \in A} Q^{\pi_{k-1}}(s,a) \quad (4)$$

Evaluation and improvement steps are iterated until convergence of π_k to π^* (which can be demonstrated to happen in a finite number of iterations when $\pi_k = \pi_{k-1}$).

2.2. Approximate Dynamic Programming

Although it seems very appealing, the method proposed above is hardly usable in real applications for two reasons. First, it assumes that the transition probabilities and the reward function are known. Practically, it is rarely feasible and especially in the case of systems interacting with humans (that would require predicting the human's behavior). Most often, only examples of interactions are available, through data collection and logging, which are actually trajectories in the state-action space. It is therefore mandatory to learn from a fixed set of data. Second, policy iteration assumes that the Q-function can be exactly represented (its value can be stored in a table for each state-action space so expression (3) represents a system of equations). However, in real world tutoring problems, state and action spaces are often too large (even continuous) for such an assumption to hold. Approximate Dynamic Programming (ADP) aims at estimating the optimal policy from trajectories when the state space is too large for a tabular representation. The Q-function is therefore approximated by some parameterized function $\hat{Q}_\theta(s,a)$ while the knowledge of the model is replaced by a database of transitions. In this paper, a linear approximation of the Q-function will be assumed:

$$\hat{Q}_\theta(s,a) = \theta^T \phi(s,a) \quad (5)$$

where $\theta \in \mathbb{R}^p$ is a vector containing the parameters and $\phi(s,a)$ is the set of p *basis functions* (or *features*). All functions that can be expressed in this way define a so-called *hypothesis space* $\mathcal{H} = \{\hat{Q}_\theta | \theta \in \mathbb{R}^p\}$. For example, $\hat{Q}_\theta$ can represent a polynomial approximation or a radial basis function network, among other possible schemes. Any function Q can be *projected* onto this hypothesis space by a projection operator Π defined as

$$\Pi Q = \operatorname*{argmin}_{\hat{Q}_\theta \in \mathcal{H}} \|Q - \hat{Q}_\theta\|^2. \quad (6)$$

The goal of the ADP algorithms is to compute the best set of parameters θ given the basis functions and the available samples. This is a problem which is more complex than supervised learning since what is observed is a set of rewards associated to transitions while the Q-function has to be approximated. In a supervised learning problem, the function to approximate is observed (sometimes with additional noise).

2.2.1. Least-Squares Policy Iteration

Least-Squares Policy Iteration (LSPI) is such an ADP algorithm [4]. LSPI is inspired by the policy iteration method and interleaves policy evaluation and improvement steps. The improvement steps are the same as before (being greedy according to the evaluated Q-function), but the evaluation step must learn an approximate representation of the Q-function using samples. In LSPI, this is done using the a modified off-policy version of the Least-Squares Temporal Differences (LSTD) algorithm [12].

LSTD aims at minimizing the distance between the approximated Q-function $\hat{Q}_\theta$ and the projection onto the hypothesis space of its image through the Bellman evaluation operator $\Pi T^\pi \hat{Q}_\theta$:

$$\theta_\pi = \operatorname*{argmin}_{\theta \in \mathbb{R}^p} \|\hat{Q}_\theta - \Pi T^\pi \hat{Q}_\theta\|^2 \quad (7)$$

This can be interpreted as trying to minimize the difference between the two sides of the Bellman equation (eq. (3)) (which should ideally be zero) in the hypothesis space. Because of the approximation, this difference is most likely to be non-zero unless Q and $\Pi T^{\pi} Q$ belong the the hypothesis space.

Practically, T^{π} is not known (we don't know the transition probabilities), but a set of N transitions $\{(s_j, a_j, r_j, s'_j)_{1 \leq j \leq N}\}$ is available. LSTD therefore solves the following optimization problem:

$$\theta_{\pi} = \underset{\theta}{\operatorname{argmin}} \sum_{j=1}^{N} C_j^N(\theta, \theta_{\pi}) \tag{8}$$

$$C_j^N(\theta, \theta_{\pi}) = (r_j + \gamma \hat{Q}_{\theta_{\pi}}(s'_j, \pi(s'_j)) - \gamma \hat{Q}_{\theta}(s_j, a_j))^2$$

Thanks to the linear parametrization, an analytical solution can be derived, which defines the LSTD algorithm:

$$\theta_{\pi} = (\sum_{j=1}^{N} \phi_j \Delta\phi_j^{\pi})^{-1} \sum_{j=1}^{N} \phi_j r_j \tag{9}$$

$$\text{with } \phi_j = \phi(s_j, a_j)$$

$$\text{and } \Delta\phi_j^{\pi} = \phi(s_j, a_j) - \gamma\phi(s'_j, \pi(s'_j))$$

LSPI therefore works as follows. An initial policy π_0 is chosen. Then, at iteration k (with $k > 1$), the Q-function of policy π_{k-1} is estimated using LSTD, and π_k is greedy respectively to this estimated Q-function. The algorithm terminates when some stopping criterion is met (e.g., small differences between consecutive policies or associated Q-functions).

3. Tutoring as an MDP

As said in the introduction, personalization of a tutoring system can be seen as a sequential decision making problem where an agent should alternate teaching and evaluation phases. The use of reinforcement learning for solving this optimization problem has already been proposed in [6] and [7]. The work presented in this paper differs by proposing a novel model but especially by using a method that learns an optimal strategy from a fixed set of data. Therefore, no interaction with learners is required during learning and any system can be improved by using simple adequate logs. To find the optimal sequence of decisions by means of reinforcement learning, one has to cast the tutoring problem into the Markov Decision Processes paradigm and thus define a set of states, a set of actions and a reward function (transition probabilities are not known but instead we have a set of data obtained from the logs of the system). Concerning actions, the casting is rather simple since there are two types of actions :

- start a teaching phase;
- start an evaluation phase.

The state representation has to contain information about the context of the interaction, that is the sufficient but necessary information required to take a decision (so as to be compliant with the Markov property). Here, the state should describe the knowledge of the student. Yet this is not directly accessible. Therefore it is modeled as a 2-dimensional vector :

- the first dimension is the correct answers rate the learner has provided so far (a continuous value between 0 and 1);
- the second dimension is the number of teaching phases the system has proposed so far (an integer).

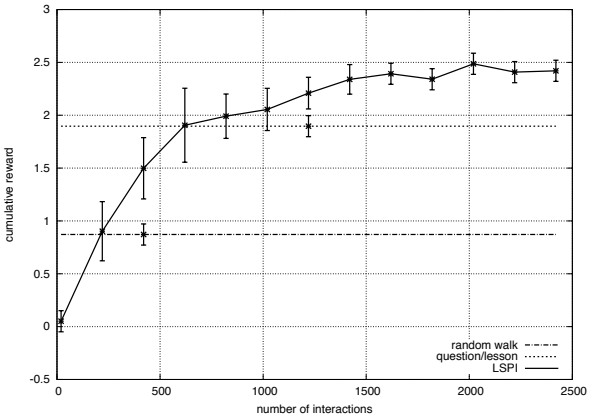

Figure 1: Result

Notice that this results in an hybrid representation of the state space (continuous/discrete) which is totally different from previous work and requires function approximation. The goal is here to adapt the system to the capabilities of learners and not to fix thresholds under which the learner is considered to fail (i.e., 90% of correct answers in the work described in [7]).

Finally, the reward is provided by the correct answer rate of the learner after each evaluation phase (a reward is obtained after each evaluation phase). Once again, the aim is to maximize this rate according to the learner's capabilities and also to accelerate the increase in the knowledge acquisition since the goal of RL is to find a strategy increasing the sequence of rewards.

4. Experiment

Because data were not available at the time of writing this article, we have simulated interactions with a learner model inspired by [13]. The model is based on a set of probabilities modeling the fact that the learner's knowledge increases or not with lessons and that s/he answers correctly to the tests by chance or not. Yet, it is important to keep in mind that this model has just been used to generate data which are compliant with our state representation but is not explicitly taken into account in the strategy learning method. From the RL algorithm point of view, everything happens as if the data were generated by real learners. Also it seemed to be a good mean to test the learnt strategy in a statistically consistent manner. The data thus take the form of logged interactions (an interaction being simply a decision of the system followed by a reaction of the user). To obtain the data, the simulated user has to interact with an initial system implementing a simple hand-crafted interaction strategy. The strategy used for data collection is a totally random policy that choses to perform evaluation or teaching phases with the same probability of 50%. The LSPI algorithm presented in Section 2.2.1 has then been applied to different data sets differing by their size. For each data set, an optimal policy has been learnt. Results are presented on Figure 1. In this figure, the value of the initial state ($V^{\pi^*}(s_0)$) (that is the discounted cumulative reward (with $\gamma = 0.9$) obtained by the learner using the learnt policy) is traced w.r.t. the number of interactions contained in the data set used by the LSPI algorithm. The goal of this experiment is to identify the number of interactions required to learn a policy that outperforms simple handcrafted policies. The performances of the learnt policies are compared to those of the random policy used for data collection and to

those of a deterministic policy that alternates teaching and evaluation phases.

It is clear on the figure that when using 500 interactions (remember that an interaction is not a complete tutoring session but simply a decision followed by the learner reaction so 500 is a quite low number in terms of tutoring sessions), the learnt policy is better that the random one (which was used to collect data). Then, after 1000 interaction, the learnt policy becomes better than the deterministic policy. So, using the logs of an existing system can be used to learn a better policy without the need of any other interaction. The obtained policy outperforms largely the one that was used to generate the data.

One can notice that those results are obtained for one type of student, modeled, as already said, by a fixed set of probabilities. The value reached by the asymptotes in the three cases as well as the gradient of the curve obtained with LSPI can thus slightly differ from one student to another. One advantage of learning tutoring strategies with LSPI is that, by construction, the best policy will make the best of each student.

So as to measure the reproducibility of the learning (e.g. the sensibility w.r.t. the dataset composition), LSPI has been run 100 times and than the learnt policy is tested 1000 times on the simulated learner. The 95% confidence interval is also plotted showing that the results are not very sensible to the randomness of the data. This is quite important since in real applications, one cannot control the quality of the data but has to use actual logs. It appears that after 1500 interactions, the confidence interval reduces and doesn't change a lot afterward.

5. Conclusion

In this paper, a method for optimizing a tutoring system is presented. This optimisation problem is first expressed as a sequential decision making problem and than solved via a reinforcement learning algorithm. Because the behavior of learners is hard to predict, a model-free method is preferred. This method only uses examples of interaction between learners and a system close to the one we want to optimize. The learnt interaction strategies are shown to outperform the basic strategies used to collect the data.

In the near future, the data collection with real students will start using the I-FLEG virtual environment. Therefore, an immediate perspective is to collect the logs of this system so as to learn optimal tutoring strategies. Also, we want to use other efficient RL algorithms [14] able to use uncertainty about Q-function parameters estimates [15] so as to improve the strategy online (while the system is used) thanks to exploration strategies that avoid disturbing the user. This method has already been successfully applied to spoken dialogue management [16].

6. Acknowledgements

The work described in this paper was conducted as part of the ALLEGRO project (www.allegro-project.eu), funded within the EU INTERREG IVa program focusing on the development of new technologies for foreign language learning.

7. References

[1] M. Amoia, C. Gardent, and L. Perez-Beltrachini, "A serious game for second language acquisition," in *Proceedings of the Third International Conference on Computer Aided Education (CSEDU 2011)*, Noordwijkerout (The Netherlands), 2011.

[2] B. S. Bloom, "Learning for mastery," *Evaluation comment*, vol. 1, no. 2, pp. 1–5, 1968.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning : An Introduction*. MIT Press, 1998.

[4] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning Research*, vol. 4, pp. 1107–1149, 2003.

[5] O. Pietquin, M. Geist, S. Chandramohan, and H. Frezza-Buet, "Sample-Efficient Batch Reinforcement Learning for Dialogue Management Optimization," *ACM Transactions on Speech and Language Processing*, 2011, accepted for publication - 24 pages.

[6] J. E. Beck, B. P. Woolf, and C. R. Beal, "ADVISOR : A machine learning architecture for intelligent tutor construction," in *Proceedings of the National Conference on Articial Intelligence*. Menlo Park, CA: MIT Press, 2000, pp. 552–557.

[7] A. Iglesias, P. Martinez, R. Aler, and F. Fernandez, "Learning teaching strategies in an adaptive and intelligent educational system through reinforcement learning," *Applied Intelligence*, vol. 31, no. 1, pp. 89–106, 2009.

[8] H. Ai, J. R. Tetreault, and D. J. Litman, "Comparing user simulation models for dialog strategy learning," in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies*, ser. NAACL-Short '07. Stroudsburg, PA, USA: Association for Computational Linguistics, 2007, pp. 1–4. [Online]. Available: http://portal.acm.org/citation.cfm?id=1614108.1614109

[9] J. Schatzmann, M. N. Stuttle, K. Weilhammer, and S. Young, "Effects of the user model on simulation-based learning of dialogue strategies," in *Proceedings of workshop on Automatic Speech Recognition and Understanding (ASRU'05), San Juan, Puerto Rico*, December 2005.

[10] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, 1957.

[11] ——, *Dynamic Programming*, 6th ed. Dover Publications, 1957.

[12] S. J. Bradtke and A. G. Barto, "Linear Least-Squares algorithms for temporal difference learning," *Machine Learning*, vol. 22, no. 1-3, pp. 33–57, 1996.

[13] A. T. Corbett and J. R. Anderson, "Knowledge tracing : Modeling the acquisition of procedural knowledge," *User modeling and user-adapted interaction*, vol. 4, no. 4, pp. 253–278, 1994.

[14] M. Geist and O. Pietquin, "Kalman Temporal Differences," *Journal of Artificial Intelligence Research (JAIR)*, vol. 39, pp. 483–532, October 2010.

[15] ——, "Managing Uncertainty within the KTD Framework," in *Proceedings of the Workshop on Active Learning and Experimental Design (AL&E collocated with AI-STAT 2010)*, ser. Journal of Machine Learning Research Conference and Workshop Proceedings, Sardinia (Italy), 2011, 12 pages - to appear.

[16] O. Pietquin, M. Geist, and S. Chandramohan, "Sample Efficient On-line Learning of Optimal Dialogue Policies with Kalman Temporal Differences," in *International Joint Conference on Artificial Intelligence (IJCAI 2011)*, Barcelona, Spain, July 2011, to appear.